ТРЕХУРОВНЕВАЯ СИСТЕМА ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ ГИБРИДНОГО 21-ЯДЕРНОГО СКАЛЯРНО-ВЕКТОРНОГО МИКРОПРОЦЕССОРА NM6408MP

Биконов Д.В.[1], Пузиков А.Д.[2], Сивцов А.С.[3], Эйсымонт Л.К.[4]

Аннотация: Гибридный микропроцессор NM6408MP (СБИС 1879ВМ8Я) с пиковой производительностью 0.5 TFlops над 32-х разрядными числами с плавающей точкой до появления Эльбрус 8С был два года самым высокопроизводительным в стране. Процессор содержит 5 ядер Cortex A5 и 16 скалярно-векторных ядер nmc4 с архитектурой NeuroMatrix. Основными языками программирования высокого уровня для процессора являются Си и Си++. Была поставлена задача разработать средства программирования для описания взаимодействия параллельных процессов и их синхронизации. В статье описываются три библиотеки разного уровня, которые решают эту задачу с учетом оригинальных особенностей процессора. В основу интерфейса пользователя функций этих библиотек взят стиль известных библиотек MPI и SHMEM.

Ключевые слова: MPI, SHMEM, сеть с коммутацией каналов, иерархическая внутрикристальная память, протокол Send/Receive, барьерная синхронизация, управление коммуникационными ресурсами процессора, эксклюзивный доступ к памяти.

DOI: *10.21681/2311-3456-2019-4-22-34*

1. Введение

Микропроцессор NM6408MP [1, 2] проектировался как высокопроизводительное и энергоэффективное средство для первичной обработки сигналов – преобразования Фурье и Адамара, фильтры, умножение плотно заполненных матриц и т.д. Впоследствии оказалось, что процессор применим еще и для решения ряда других задач, например, видеообработки и нейровычислений.

Многоядерность процессора потребовала реализации для него современных средств параллельного программирования. Для начала были выбраны небольшие подмножества функций библиотек MPI [3] и SHMEM[4].

NM6408MP достаточно экономно спроектирован, даже аскетически экономно, поскольку предполагалось его использование во встроенных системах разного типа. Это отличает его от современных зарубежных многоядерных процессоров. По этой причине при реализации выбранных средств параллельного программирования пришлось преодолеть ряд трудностей необычного характера, что потребовало пойти на определенное усложнение программного обеспечения, которое привело к некоторому снижению базовых характеристик в сравнении с возможностями аппаратуры по пропускным способностям передачи сообщений, временам выполнения синхронизаций. По-существу, при реализации средств параллельного программирования пришлось реализовать систему контроля и управления коммуникационными ресурсами процессора. Так появилась рассматриваемая в статье библиотека БУПВ/М.

Однако оказалось, что некоторые пользователи готовы в определенной степени потерять удобные и гарантированно надежные возможности ради достижения характеристик, вплотную приближающихся к возможностям аппаратуры. Так появились две библиотеки нижнего уровня, сначала БФСКН, а потом, более экстремальная, – БФСКН-Ф.

В разделе 2 статьи объясняются архитектурные особенности NM6408MP и специфика разработки библиотек. Базовые возможности библиотек и решения по реализации перечислены в разделе 3. Потом, в разделе 4, перечисляются функции этих библиотек и показывается их применение на примере пересылки сообщения. В конце этого раздела приводятся характеристики функций пересылки сообщений и барьерной синхронизации, полученные на реальном оборудовании.

Заключение содержит определенные обобщения и планы работ, которые связаны с обеспечением параллельного программирования уже на сети микропроцессоров NM6408MP.

1 Биконов Дмитрий Владиленович, главный специалист ЗАО «НТЦ «Модуль» , г. Москва, Россия. E-mail: d.bikonov@module.ru
2 Пузиков Артем Дмитриевич, инженер-программист ЗАО «НТЦ «Модуль» , г. Москва, Россия. E-mail: a.puzikov@module.ru
3 Сивцов Алексей Сергеевич, старший инженер-программист ЗАО «НТЦ «Модуль» , г. Москва, Россия. E-mail: a.sivtsov@module.ru
4 Эйсымонт Леонид Константинович, к.ф.-м.н., научный консультант ЗАО «НТЦ «Модуль» , г. Москва, Россия. E-mail: verger-lk@yandex.ru

УДК 004.45+004.272.43 DOI: 10.21681/2311-3456-2019-4-22-34

2. Особенности микропроцессора NM6408MP

Структура NM6408MP в упрощенном виде представлена на рис.1. Это иерархия узлов типа ARM (ядро Cortex A5 с локальной памятью) и типа NMPU (скалярно-векторное ядро nmc4 с локальной памятью, см. рис.2). Структура процессора ориентирована на модель организации параллельной программы типа »master-slave». Имеется центральный управляющий ARM-узел и четыре кластера, каждый из кастеров содержит управляющий ARM-узел и четыре NMPU-узла.

Все узлы процессора связаны через древовидную среднескоростную сеть AXI передачи пакетов и высокоскоростную двухуровневую сеть коммутации каналов. Обе сети подключены к центральному узлу. В сети коммутации каналов на первом уровне кластеры соединены по типу »каждый с каждым», а на втором уровне в каждом кластере таким же образом связаны NMPU-узлы.

Имеются пять портов внекристальной DDR-памяти и четыре EL-линка для межкристальных взаимодействий, которые позволяют строить сети из таких процессоров.

Дополнительно центральный ARM-узел имеет интерфейс PCI-e 2.0, что позволяет подключиться к host-процессору непосредственно, либо через коммутатор PCI-экспресс подключить несколько процессоров к одному host-процессору.

NMPU-узел – основная особенность процессора. Он включает четыре конвейерных устройства VALUi для выполнения векторных операций над 32-х и 64-х разрядными числами с плавающей точкой (FP32 и FP64), а также специальное устройство структурного преобразования векторов – Pack-Unpack Unit. Каждое VALUi может за такт выполнять 8 операций над FP32 (вычисление D0 и D1 на рис.2) и до 2-х операций над FP64 (вычисление D на рис.2).

Каждое VALUi непосредственно работает с 8-ю векторными регистрами, каждый регистр – 32 элемента по 64 разряда. VALUi вместе с векторными регистрами образует «векторную ячейку» (Cell), эти ячейки могут передавать друг другу вектора через коммутатор, минуя обращения к памяти. Выдача команд в векторные ячейки производится RISC-процессором, входящим в состав NMPU-узла.

NMPU-узел еще содержит состоящую из 8 блоков локальную память с высокой пропускной способностью, подключенную через специальный блок, называемый системным интегратором (System Integrator), который содержит коммутаторы адресов и данных, 8 генераторов адресов для доступа к элементам векторов в локальной памяти по векторным командам считывания/записи векторов.

Отметим, что по архитектурному решению NMPU-узел соответствует определенным современным тенденциям развития архитектур в мире. Например, он похож на DSP-ядро китайского GPDSP FT-Matrix2000 [5, 2], который разрабатывался как процессор-ускоритель для экзафлопсного суперкомпьютера Tianhe-3 и должен выполнять в нем такую же роль, как GPGPU фирмы NVIDIA в американских суперкомпьютерах высшего диапазона производительности. Сравнение NMPU-узла с DSP-ядром имеется в работе [2].

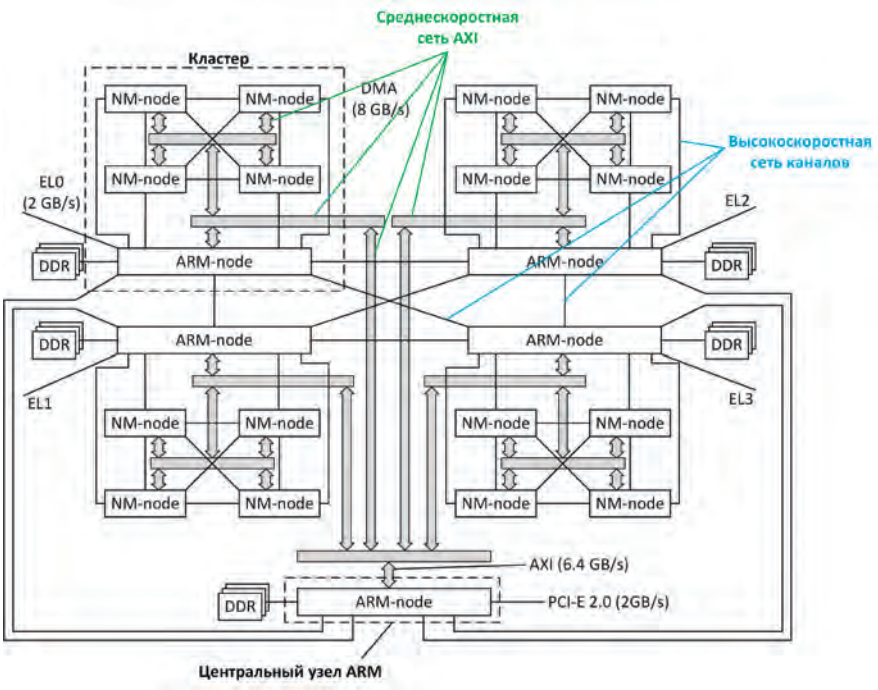

Рис.1. Структура микропроцессора NM6408MP

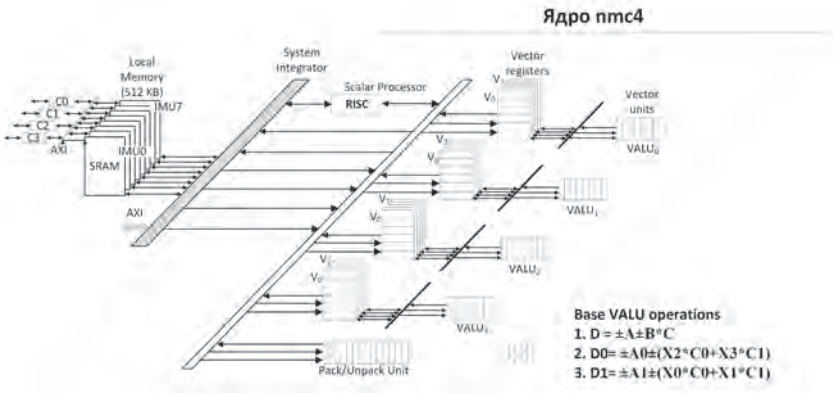

Рис.2. Структура узла NMPU

Далее перейдем к главной теме при рассмотрении любого процессора или суперкомпьютера – к вопросу организации памяти и работе с ней.

Узлы NM6408MP вместо иерархии кэш-памятей данных, как это обычно принято, имеют локальные памяти, которые адресуются через единое адресное пространство процессора, доступ реализуется через сеть AXI.

На рис.3 схематически представлена иерархия памятей узлов с указанием количества тактов, за которые происходят обращения от ядер к памятям этой иерархии. Кроме того, на рисунке приведены пропускные способности пересылки данных с использованием векторных операций считывания. Из рисунка видно, что обращения к памятям других узлов имеют большие задержки выполнения в сравнении с локальной памятью узла. Иными словами, процессор – это ярко выраженная система с неоднородным доступом к памяти (NUMA-система).

Для групповых пересылок по AXI-сети кроме векторных операций еще можно использовать специальные блоки MDMAC пересылки данных типа »память-память». Их имеется пять штук, по одному на каждый кластер и один для центрального узла.

Отказ от использования кэш-памятей данных в узлах процессоров и замена их на быстрые адресуемые памяти (блокнотные, Scratchpad памяти) наблюдается во многих разработках, но только если он применяется в сочетании с механизмами асинхронной загрузки/выгрузки данных. Для реализации асинхронности требуются DMA-устройства и быстрые каналы передачи данных [8, 9]. Наличие таких коммуникационных ресурсов – вторая особенность NM6408MP после введенной в NMPU-узлах обработки векторов. Коммуникационные ресурсы NM6408MP: 84 полнодуплексных компортов DMA (»память-канал»|); 16 коммутаторов каналов компортов (3-х и 4-х портовых);

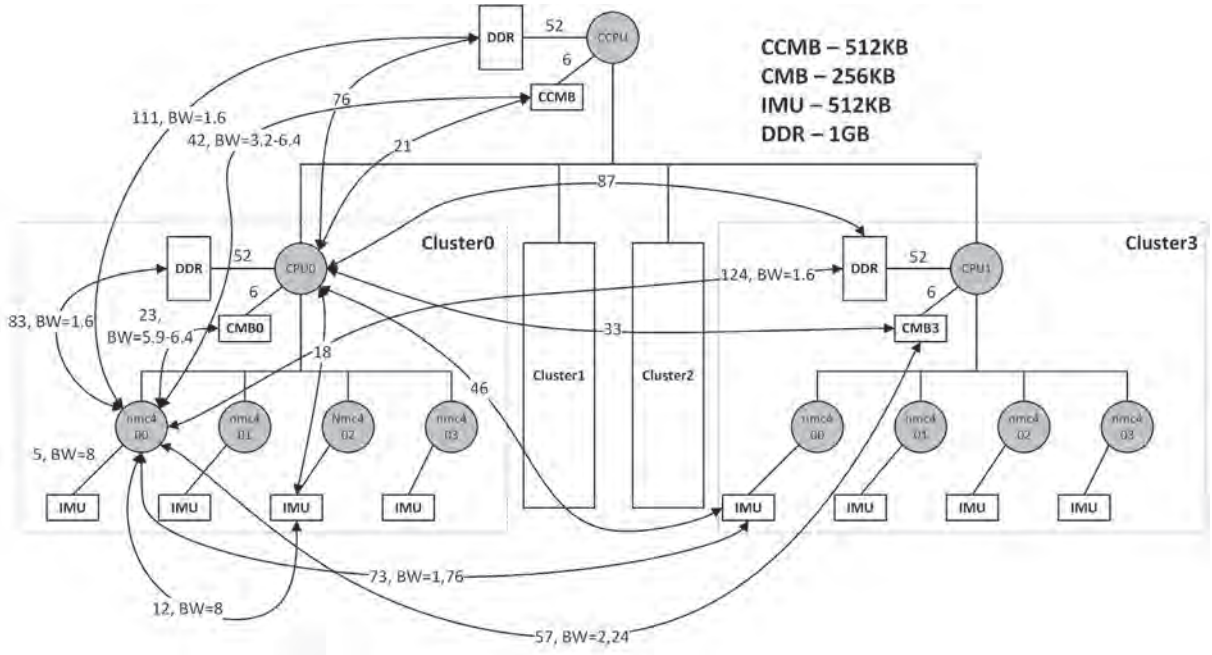

Рис.3. Иерархия памятей узлов процессора NM6408, задержки обращений к памяти в тактах, пропускные способности при векторном доступе (BW, GB/s) из ядер nmc4.

УДК 004.45+004.272.43 DOI: 10.21681/2311-3456-2019-4-22-34

В сети коммутации каналов настройка маршрутов передачи сообщений должна программироваться посредством задания значений ее управляющих регистров. Интересно, что такую организацию внутрикристальной сети с начала текущего тысячелетия было предложено не использовать [6], а использовать сеть с коммутацией пакетов с более сложным оборудованием, но предоставляющей большие удобства программистам. Тем не менее, судя по приведенным в данной статье результатам, а также появлением программно реконфигурируемых сетей коммутации каналов в новейших процессорах для обработки графов [7], мнение о »правильной» внутрикристальной сети в современных условиях должно быть пересмотрено.

Перейдем к более подробному рассмотрению вопросов построения маршрутов в сети коммутации каналов процессора. Это первая главная проблема при реализации библиотек параллельного программирования, вторая проблема – обход ограничений в NM6408MP реализации атомарных операций с памятью через команды эксклюзивного чтения/записи, принятых в ARM-архитектуре [12].

На рис.4 дана схема сети коммутации каналов, в которой указаны компорты DMA (CPj-соединения на рис.4) и коммутаторы компортов (LCij-блоки на рис.4), а также показаны примеры трех типичных маршрутов: CP-CP – прямое соединение двух компортов (CP3 узла NMPU00 и CP3 узла NMPU03) ; CP-SW-CP – соединение двух компортов через соответствующим образом настроенный коммутатор (CP0 узла NMPU01 и CP1 узла CPU0 соединяются через коммутатор LC01, в котором настроено соединение от порта1 к порту 2); CP-SW-SW-CP – соединение двух компортов через два коммутатора (CP0 узла NMPU03 и CP0 узла NMPU33 соединяются через коммутатор LC03 с настроенным соединением от порта 1 к порту 0 и коммутатор LC33 с настроенным соединением от порта 0 к порту 1).

Настройка ресурсов маршрутов может быть выполнена пользователем, а может сделана автоматически по заданию узлов (или процессов) начала и конца маршрута. В библиотеках параллельного программирования допускаются оба подхода.

Рассматривая схему соединения узлов рис.4, можно увидеть, что не для любых двух узлов процессора можно построить маршрут. Например, нет прямого маршрута через компорты и коммутаторы между узлом NMPU03 и NMPU23. Такие маршруты не были предусмотрены в процессоре, поскольку он разрабатывался под модель «master-slave». Однако при разработке библиотек было принято решение для общности такие маршруты также реализовать за счет специального программного обеспечения. Такие маршруты были названы X-маршрутами, задача их реализации – «проблемой X-маршрутов». В существующих версиях библиотек пока эти маршруты не реализованы, хотя алгоритмы реализации проработаны.

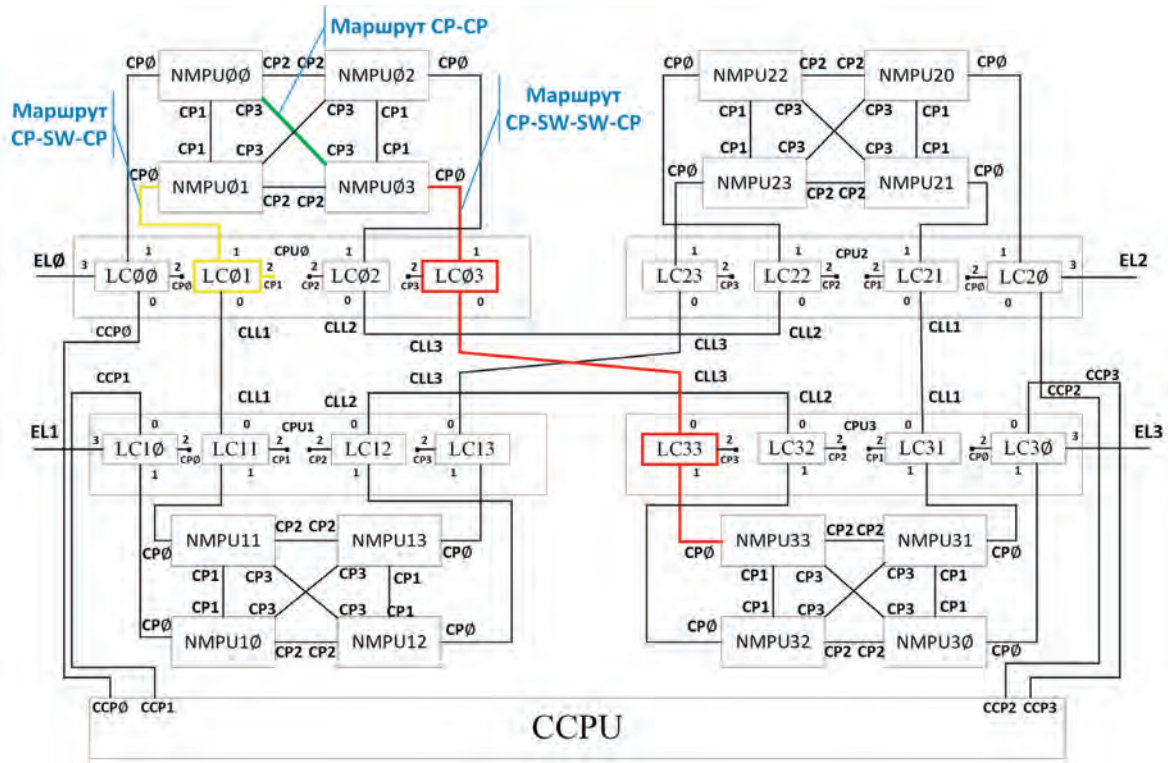

Рис.4. Схема соединения узлов NM6408 через компорты и коммутаторы компортов сети коммутации каналов

При реализации маршрутов используется возможность работы с управляющими регистрами коммуникационных ресурсов через общее адресное пространство, аппаратно обеспеченное в одном процессоре. Если расширять возможности библиотек до работы на сети процессоров NM6408MP, то надо учитывать, что в каждом процессоре адресное пространство свое, поэтому из одного процессора обратиться в другой процессор по логическому адресу не удастся. Реализация взаимодействия в сети процессоров обозначается как общая «проблема EL-линков», от названия внешних EL-линков, через которые процессоры соединяются, см. рис.1. Алгоритмы решения этой проблемы прорабатываются в настоящее время.

Последняя проблема, связанная с особенностями NM6408MP, которую пришлось решать при разработке библиотек, – ограничения реализации эксклюзивных команд чтения и записи [12]. Проблема состоит в следующем.

Для реализации команд эксклюзивного доступа в контроллерах локальных памятей ARM-узлов введены специальные блоки – мониторы, по 6 штук в каждом. Предположим для простоты, что один процесс может одновременно работать только с одним монитором, т.е. в этом процессе последовательно не могут быть выданы более одной команды эксклюзивного доступа по разным адресам. Тогда получается, что внутри одного кластера, либо на уровне центрального узла могут выполняться до 6 параллельных процессов, по количеству имеющихся мониторов. Внутри кластера это могут быть, например, процессы от 4-х узлов NMPU, ARM-узла кластера и центрального узла. На уровне центрального узла – это могут быть ARM-узлы кластеров, сам центральный узел и host-процессор. Такое статическое распределение прав логично, если параллельная программа будет иметь строго модель «master-slave».

Для более общей модели, когда процесс из одного кластера может потребовать права эксклюзивного доступа к ресурсам памяти в другом кластере, требуется специальное решение с динамическим предоставлением прав, а не таким простым статическим, как описано выше. Эта проблема получила название «проблема мониторов» и в текущих версиях библиотек решена.

3. Базовые возможности библиотек параллельного программирования

Библиотеки БФСКН-Ф и БФСКН являются библиотеками функций синхронизации и коммуникаций нижнего уровня. Для достижения быстродействия в их функциях не обеспечивается контроль использования коммуникационных ресурсов.

БФСКН-Ф – обеспечивается возможность лишь ручной статической установки маршрутов с непосредственной настройкой компортов и коммутаторов, маршруты прокладываются между физическими ядрами процессора.

БФСКН – обеспечивает возможность статической автоматической установки маршрутов в сети компортов и коммутаторов между параллельными процессами, которые в соответствии с заданием пользователя размещаются на физических ядрах процессора.

БУПВ/М – библиотека высокого уровня, обеспечивает полный контроль занятости коммуникационных ресурсов, автоматическое статическое и динамическое (непосредственно перед пересылкой сообщения) построение маршрутов между процессами, работу с событиями и сигналами, mutex-переменными.

Детализация возможностей библиотек дана в таблице 1, в следующем разделе их функции уточняются. Отмеченные звездочкой возможности находятся в процессе реализации.

Все библиотеки работают в модели «одно ядро-один процесс» со статическим размещением процессов пользователя на ядрах перед выполнением программы. Считается, что на всех узлах (кроме центрального) нет операционной системы.

Привязка процессов программы пользователя к физическим ядрам процессора задается пользователем в задании на загрузку программы host-процессору. Каждый процесс пользователя имеет уникальный номер, который и используется в параллельных программах, если применяются библиотеки БФСКН или БУПВ/М. Соответствие номеров процессов физическим номерам ядер и внутренним «сквозным номерам» хранится в этих библиотеках в специальных таблицах TFX, которые строятся при инициализации параллельной программы по заданию на загрузку от пользователя и хранятся в каждом загруженном программой узле все время ее выполнения.

Таблица 1

Сравнительные возможности трех библиотек параллельного программирования

Возможности	БФСКН-Ф	БФСКН	БУПВ/М
Физическая адресация процессов через адреса выполняющих их узлов процессора	+		
Физическая адресация компортов	+		
Физическая адресация коммутаторов компортов	+		
Явная установка соединений в коммутаторах компортов	+		
Логическая адресация процессов		+	+
Управляемое размещение процессов на узлах процессора		+	+
Автоматический контроль занятости ресурсов			+

Автоматическая поддержка протоколов работы с ресурсами удаленного кластера			+
Автоматическая статическая прокладка маршрутов		+	+
Автоматическая динамическая прокладка маршрутов			+
Автоматический захват свободных блоков пересылки данных MDMAC		+	+
Работа с mutex-переменными		+	+
Работа с сигналами			+
Работа с событиями			+
Локальная барьерная синхронизация	+	+	+
Глобальная барьерная синхронизация	+	+	+
Автоматическая поддержка X-маршрутов (через теневые процессы)			+(*)
Автоматическая поддержка EL-маршрутов (X-маршруты для сети процессоров)		+	+(*)
Трассировка выполнения и взаимодействия процессов			+(*)
Встроенный механизм обработки прерываний от коммуникационных устройств			+
Пересылка сообщения по маршруту в сети коммутации каналов	+	+	+
Автоматическое вычисление адресов в глобальном адресном пространстве процессора и обращения по ним посредством операций PUT и GET			+

Для автоматической прокладки маршрутов в библиотеках БФСКН и БУПВ/М имеется таблица TRS всех маршрутов между физическими ядрами процессора размером 21x21 элемент, в соответствии с имеющимися у физических ядер сквозными номерами.

Общая схема автоматической прокладки маршрута следующая. Если требуется передача сообщения из процесса с номером A в процесс с номером B, то по таблице TFX сначала определяются «сквозные номера» физических ядер, на которых процессы размещены, пусть это будут N_A и N_B. Далее из элемента (N_A, N_B) таблицы TRS выбирается зашифрованный маршрут в виде требуемых настроек компортов и коммутаторов. В зависимости от типа маршрута, его реализует далее соответствующая программа-исполнитель из состава встроенных в библиотеку.

Если требуется не только автоматическое построение маршрутов, но и контроль использования коммуникационных ресурсов, чтобы не возникло конфликтующих маршрутов, введены возможности отслеживания запущенных процессов пересылок (фреймы процессов пользователя и системных процессов пересылок), а также средства отслеживания использования коммуникационных ресурсов (дескрипторы компортов и коммутаторов, блоков MDMAC). Работа с фреймами и дескрипторами скрыта от пользователя, все это реализовано только в библиотеке БУПВ/М.

При пересылке сообщений по сети коммутации каналов используется протокол Send/Receive, как в библиотеке MPI [3], поскольку настройка коммуникационных компортов процесса-поставщика и процесса-получателя может выполняться только узлами, в которые эти компорты входят.

Проблема мониторов в БУПВ/М решается встроенными турнирными схемами захвата процессами прав доступа к ресурсам, которые всегда гарантируют выполнение не более 6 параллельных процессов в кластере или на уровне центрального узла.

Проблема X-маршрутов будет решена в БУПВ/М введением теневых процессов и использующих их специальных протоколов пересылки сообщений.

Проблема EL-маршрутов будет решена в БУПВ/М введением механизмов X-маршрутов, специальных сообщений и протоколов передачи сообщений в ARM-узлах с подключенными EL-линками.

4. Функции библиотек и характеристики

В разделах 4.1-4.3 дается перечень имен функций библиотек, примеры описания передачи сообщения по маршруту CP-SW-SW-CP (NMPU03 – NMPU33, рис.3) и списки преимуществ и недостатков. В разделе 4.4 имеются экспериментальные данные (пропускная способность и задержки) по пересылке сообщений разной длины через сеть коммутации каналов, а также через блоки MDMAC и сеть AXI, времена барьерной синхронизации для разных множеств ядер процессора.

4.1 Библиотека БФСКН-Ф

БФСКН-Ф является библиотекой самого низкого уровня. Она работает непосредственно с ядрами и коммуникационными ресурсами по физическим номерам. Эту библиотеку можно представить как API к регистрам коммуникационных ресурсов процессора.

В библиотеку БФСКН-Ф входят следующие функции, их имена достаточно ясно соответствуют назначению функций, а также дополнительно прокомментированы на примере:

1. Функции барьерной синхронизации – local_barrier, global_barrier, arm_barrier, nm_barrier.
2. Функции работы с коммутаторами и компортами – set_send_cp, set_recv_cp, set_lc, start_send, start_recv, check_cp, wait_send_cp, wait_recv_cp.
3. Функции работы с блоками пересылки данных – start_mdmac, check_mdmac, wait_mdmac.

Пример 1. Пересылка сообщения из NMPU03 в NMPU33 (рис. 4). На стороне процесса-поставщика сначала соединяются порты 1 и 0 в коммутаторе LC0. Потом компорт CP0 настраивается на 2-мерную адресацию с доступом подряд к четырем словам, потом пропуск четырех слов и т.д., так формируется передаваемое сообщение. После этого запускается пересылка через CP0 сообщения длиной 8192 двойных слов с адреса 0x10000. В последнем операторе производится ожидание окончания работы компорта CP0 по выдаче сообщения. На стороне процесса-потребителя действия похожие, только производится настройка на прием сообщения.

Процесс-поставщик в узле NMPU03

```
set_lc (LC03, 1, 0);
set_send_cp (CP0, _2D, 0x4, 0x4);
start_send (CP0, 0x10000, 8192);
wait_send_cp (CP0);
```

Процесс-получатель в узле NMPU33

```
set_lc (LC33, 0, 1);
set_recv_cp (CP0, _2D, 0x4, 0x4);
start_recv (CP0, 0x10000, 8192);
wait_recv_cp (CP0);
```

Операторы ожидания окончания работы компортов можно использовать не сразу после запуска передачи или приема. Между ними могут быть вычислительные операторы процессов, они тогда будут выполняться одновременно с выполнением пересылки сообщения.

Для того, чтобы в процессе-поставщике узнать о приеме сообщения в процессе-получателе, необходимо предпринять специальные действия по уведомлению процесса-поставщика от процесса-получателя. В данном примере это не указано. □

Библиотека БФСКН-Ф имеет следующие преимущества:

1. Минимальные задержки передачи коротких сообщений из-за принципиального использования статически построенных маршрутов.
2. Ожидаемая хорошая крутизна роста пропускной способности от длины передаваемого сообщения (см. раздел 4.4) и близкая к пиковой пропускная способность на длинных сообщениях.
3. Отсутствие необходимости хранения служебных данных библиотеки в памяти процессора.

Недостатки библиотеки БФСКН-Ф:

1. Пользователю необходимо знать топологию сети коммутации каналов и помнить о жестком размещении своих пользовательских процессов на узлах (рис.4).
2. Если программа занимает не все узлы, а те, что занимает, сбоят или отказали, то для переноса программы на другие узлы ее надо значительно переделать.
3. Пользователю необходимо помнить об использовании коммуникационных ресурсов, чтобы не было конфликтов маршрутов.
4. Отсутствует решение проблемы мониторов, пользователь должен об этом заботиться сам.
5. Отсутствуют средства передачи сообщений между узлами процессора, где нет прямого соединения (проблема X-маршрутов).
6. Отсутствуют средства работы в сети процессоров (проблема EL-линков)

4.2 Библиотека БФСКН

Данная библиотека отличается от БФСКН-Ф тем, что в ней реализована работа с процессами по их логическим номерам, вводится автоматическое построение маршрутов, решается проблема мониторов для функций работы с mutex-переменными (для этого даже введены упрощенные фреймы процессов пользователя). Плата за работу с процессами по их логическим номерам и автоматическое построение маршрутов – появление таблиц TFX и TRS, т.е. захват некоторого объема памяти у программы пользователя.

В библиотеку БФСКН входят следующие функции:

1. Функции барьерной синхронизации – local_barrier, global_barrier, arm_barrier, nm_barrier.
2. Функции работы с коммутаторами и компортами – send_path, recv_path, send_path_on, recv_path_on, check_send, wait_send, check_recv, wait_recv.
3. Функции работы с блоками пересылки данных – start_mdmac, check_mdmac, wait_mdmac
4. Функции работы с mutex-переменными – lock, unlock.

Пример 2. Пересылка сообщения из NMPU03 в NMPU33 (рис. 4). Будем считать, что процесс-поставщик имеет логический номер 2, а процесс-получатель – логический номер 4.

Процесс-поставщик в узле NMPU03

```
send_path_on (4, _2D, 0x4, 0x4);
send_path (4, 0x10000, 8192);
wait_send (4);.
```

Процесс-получатель в узле NMPU33

```
recv_path_on (2, _2D, 0x4, 0x4);
recv_path (2, 0x10000, 8192);
wait_recv (2);
```

В этом примере устанавливается маршрут из процесса 2 в процесс 4, про коммутаторы и компорты на данном пути не упоминается, их настраивают программы библиотеки. □

Библиотека БФСКН имеет следующие преимущества:

1. Ожидаемые относительно невысокие задержки передачи коротких сообщений из-за использования статически построенных маршрутов.
2. Ожидаемая хорошая крутизна роста пропускной способности от длины передаваемого сообщения и близкая к пиковой пропускная способность на длинных сообщениях.
3. Отсутствие необходимости знать топологию соединения ядер процессора через коммуникационные ресурсы.
4. Переход на использование новых узлов, если некоторые из узлов процессора отказали, производится без переделки программы.

5. Решается проблема мониторов для функций работы с mutex-переменными.
6. Есть принципиальная возможность решения проблемы X-маршрутов и EL-маршрутов, поскольку используется автоматическое построение статических маршрутов.

Недостатки библиотеки БФСКН.

1. Наличие служебных данных, занимающих память процессора, хотя большие таблицы TFX и TRS могут быть размещены в DDR-памяти (по заданию от пользователя) без существенной потери производительности, поскольку применяются только статические маршруты.
2. Пользователю необходимо помнить об использовании коммуникационных ресурсов, чтобы не было конфликтов маршрутов, поскольку нет контроля их использования.
3. Используется передача сообщений только по статическим маршрутам.
4. Принципиально возможен, но сильно затруднен из-за отсутствия системы контроля использования ресурсов запуск множества асинхронных процессов обмена для одного процесса пользователя.

Видно, что возможностей у БФСКН по отношению к БФСКН-Ф гораздо больше и она лишена большей части недостатков, остался лишь недостаток по отсутствию контроля использования коммуникационных ресурсов процессора.

4.3 Библиотека БУПВ/М

Библиотека БУПВ/М (М – означает модернизированная) – библиотека самого высокого уровня. Она приближена к базовым возможностям библиотек MPI и SHMEM. В этой библиотеке значительно расширена функциональность (см. Таблицу 1), введена работа с событиями и сигналами, разрешена передача по динамически прокладываемым маршрутам, введены такие элементы SHMEM, как прямое обращение к данным иерархической памяти (см.рис.3) через глобальное адресное пространство.

Если же говорить о реализации БУПВ/М, то введен контроль использования коммуникационных ресурсов и контроль запуска асинхронных процессов обмена, проблема мониторов решена для всех функций, оптимизировано размещение системных переменных синхронизации (большая их часть находится в локальных памятях кластерных ARM-узлов, а не в центральном узле).

В библиотеку БУПВ/М входят следующие функции:

1. Функции барьерной синхронизации – MP_local_barrier, MP_global_barrier, MP_arm_barrier, MP_nm_barrier.
2. Функции передачи данных – MP_send, MP_recv, MP_send_path_on, MP_send_path_off, MP_send_path_check, MP_send_path, MP_recv_path_on, MP_recv_path_off, MP_recv_path_check, MP_recv_path
3. Функции работы с блоками пересылки данных – MP_move
4. Функции работы с сигналами – MP_send_signal, MP_broadcast_signal, MP_wait_signal, MP_wait_any_signal, MP_drop_signal
5. Функции работы с событиями – MP_set_event, MP_wait_event, MP_drop_event
6. Функции работы с mutex-переменными – MP_lock, MP_unlock
7. Функции работы в глобальном адресном пространстве – MP_global_address, MP_put, MP_get,
8. Разные вспомогательные и сервисные функции – MP_init, MP_finalize, MP_my_ID, MP_check_send_wait, MP_check_recv_wait, MP_check_extrn_send_wait, MP_check_extrn_recv_wait, MP_drop_send_wait, MP_drop_recv_wait

Функции MP_send и MP_recv предназначены для пересылки сообщений с непосредственно перед пересылкой установкой маршрута, т.е. в динамике. Процесс-поставщик, выполняя MP_send, сначала строит свою часть маршрута до процесса-получателя и подготавливается к началу процесса пересылки. Построить весь маршрут нельзя из-за ограничений в работе аппаратных средств. Процесс-потребитель по MP_recv устанавливает свою часть маршрута и подготавливается к получению сообщения. Как только обе части маршрута построены, процесс-поставщик об этом узнает и начинает собственно пересылку сообщения. Она происходит независимо (асинхронно) от выполнения программы в процессе-поставщике и процессе-получателе для сообщений любой длины. Это преимущество сети с коммутации каналов.

Окончание выдачи сообщения из процесса-поставщика фиксируется по окончанию работы его компорта и отражается в событийной переменной, заданной пользователем в обращении к MP_send.

Окончание приема сообщения в процессе-получателе фиксируется по окончанию работы компорта на прием и отражается в соответствующей событийной переменной, заданной пользователем в обращении к MP_recv. После этого процесс-получатель сигнализирует процессу-поставщику о полной передаче сообщения в еще одной событийной переменной, заданной пользователем уже в обращении в этом процессе к MP_send.

Остальные функции группы функций передачи данных предназначены для установки статических маршрутов в сети коммутации каналов и последующей передачи сообщений по ним. Если маршрут построен функциями MP_send_path_on и MP_recv_path_on, то далее по нему можно переслать сообщения произвольное количество раз функциями MP_send_path и MP_recv_path.

Если установленный статический маршрут не будет далее использоваться, то его ресурсы могут быть освобождены функциями MP_send_path_off и MP_recv_path_off. Проверка наличия фрагментов маршрутов на передачу и прием в процессе может быть выполнена функциями MP_send_path_check и MP_recv_path_check.

В списке разных вспомогательных функций есть очевидные по назначению функции инициализации и завершения использования библиотеки БУПВ/М, что типично для MPI. Имеется функция определения из текущего процесса его логического номера – MP_my_ID. Имеется еще ряд функций проверки «висящих»(без соответствующего Send или Recv) обращений передачи и приема сообщений с динамической прокладкой маршрутов. Эти функции введены

для обеспечения возможности самоконтроля параллельных программ.

Пример 3. Пересылка сообщения из NMPU03 в NMPU33 (рис. 4). Пусть также процесс-поставщик имеет номер 2, а процесс-потребитель – номер 4. Показывается передача сообщения без предварительной прокладки маршрута.

Процесс-поставщик в узле NMPU03

```
MP_send (4, send_bufer, N,
                &V1,&V2,&V3,&V4);
```

Процесс-получатель в узле NMPU33

```
MP_recv (2,recv_bufer,N,
                &V1,&V2,&V3);
```

В обращениях к функциям имеются переменные V1, ...V4, они нужны для диагностики разных событий и фиксации разных этапов выполнения функций. Программа пользователя может работать с этими переменными, если требуется повышенная надежность, либо не работать.

Для MP_send назначение этих переменных следующее:

V1 – информирует, возможно ли обращение к MP_send и правильно ли заданы параметры обращения;

V2 – информирует об ошибках в процессе выполнения функции MP_send;

V3 – событийная переменная, устанавливается при окончании выдачи сообщения из процесса-поставщика;

V4 – событийная переменная, устанавливается после окончания приема сообщения в процессе-потребителе.

Для MP_recv переменные V1 и V2 имеют такое же назначение, что и V1 и V2 для MP_send, а V3 – событийная переменная, устанавливается после окончания приема сообщения в процессе-потребителе. □

4.4. Результаты оценочного тестирования библиотек

Далее приведены результаты исследования пропускных способностей и задержек передачи сообщений разной длины между параллельными процессами через высокоскоростную сеть коммутации каналов, а также через блок MDMAC и среднескоростную сеть AXI. Дополнительно приводятся данные по временам барьерных синхронизаций.

Это минимальный набор характеристик библиотек параллельного программирования. Для сравнения используются результаты измерений на 4-х сокетных платах с микропроцессорами AMD Magny Cours (2.2 GHz), соединенных сетью типа «каждый с каждым» через каналы межкристального взаимодействия HyperTransport (пиковая пропускная способность 6.4 GB/s, 8-битовые линки) [10, 11].

Результаты измерения пропускных способностей для сообщений, передаваемых через сеть коммутации каналов, приведены на рис.5, отметим следующее.

1. Пропускные способности библиотек БФСКН-Ф и БФСКН очень близки, как для пересылок между NMPU-узлами, так и для АРМ-узлов.
2. Пропускные способности библиотек БУПВ/М отстают и составляют от 30% (короткие сообщения) до 90% (длинные сообщения). Возможны две причины: накладные расходы из-за динамической прокладки маршрута непосредственно перед пересылкой сообщения; неоптимизированный код реализации соответствующей функции библиотеки. Первая причина, скорее всего, более значима и может быть снята за счет использования функций БУПВ/М с предварительной статической прокладкой маршрута.

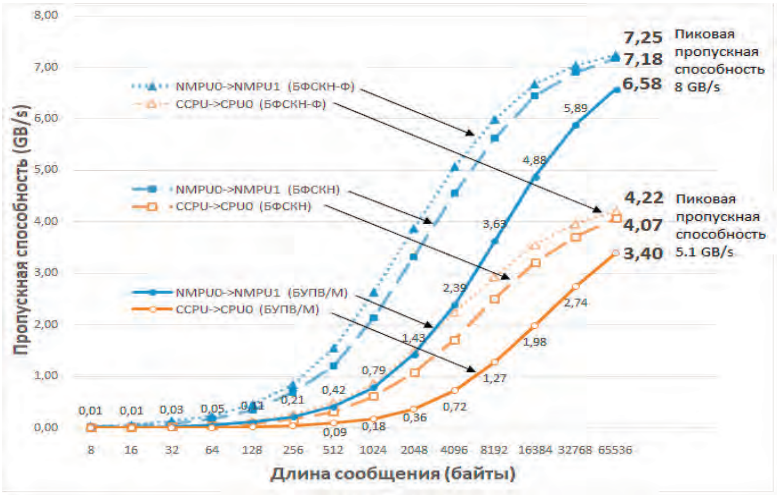

Рис.5. Пропускные способности передачи сообщений между узлам через компорты с использованием разных библиотек

3. Для библиотек БФСКН-Ф и БФСКН на длинном сообщении в 65536 байт при пересылке между NMPU узлами достигается 90.6% пропускной способности от пиковой, а при пересылке между центральным узлом и кластерным ARM – 82.4% от пиковой.
4. Для библиотек БФСКН-Ф и БФСКН половина пиковой пропускной способности (так определяем характеристику »крутизна») достигается при пересылке между NMPU-узлами на сообщении в 2048 байтов, а при пересылке между центральным узлом и кластерным ARM – на сообщении в 4096 байтов.

В работе [11] приведены более скромные результаты аналогичного тестирования 4-х сокетной платы с микропроцессорами AMD MagnyCours. На сообщении в 65536 байтов достигается лишь 26.5% пропускной способности от пиковой, а остановка роста пропускной способности возникает лишь на сообщении длиной 262144 байта и достигается значение в 45.3 % от пиковой. Причина такого отставания по характеристикам в общности и громоздкости реализации MPI, отличающейся множеством копирований сообщений в памяти из-за использования буферов для приема сообщений из сети передачи пакетов (для канала можно сразу указать область памяти, куда это сообщение должно быть принято).

Результаты измерения задержек для сообщений разной длины, передаваемых через сеть коммутации каналов, приведены на рис.6, отметим следующее.

3. Для передач между NMPU-узлами на библиотеках БФСКН-Ф и БФСКН получены рекордно малые задержки уровня 0.25 мксек, а для БУПВ/М получена задержка в 1 мксек, что на уровне обычно получаемых задержек для реализаций MPI [10, 11], но эта характеристика может быть еще улучшена.

На рис.7 и 8 приведены пропускные способности и задержки для пересылок сообщений через блоки MDMAC и среднескоростную сеть AXI. Эти результаты получены для библиотек БФСКН-Ф и БФСКН, для БУПВ/М измерения не производились.

Блоки MDMAC выполняют пересылку »память-память», в этом плане они сложнее компортов, работают всегда с AXI-шинами в 800 MHz и 640 MHz уровня ARM-узлов кластеров и центрального узла. Это ограничивает их пиковую пропускную способность соответственно в 6.4 GB/s и 5.1 GB/s.

При пересылке по AXI-шине в 800 MHz между блоками памяти ARM-узла достигается 89.2% от пиковой пропускной способности на сообщении длиной 65536 (здесь и далее) при хорошей крутизне. Этот процент не хуже, чем для компорта, хотя MDMAC сложнее.

При пересылке из DDR-памяти в память ARM-узла через AXI-шину 800 MGz достигается 75% пропускной способности от пиковой. Это говорит о степени согласованности по пропускной способности DDR-памяти, AXI-шины и памяти ARM-узла.

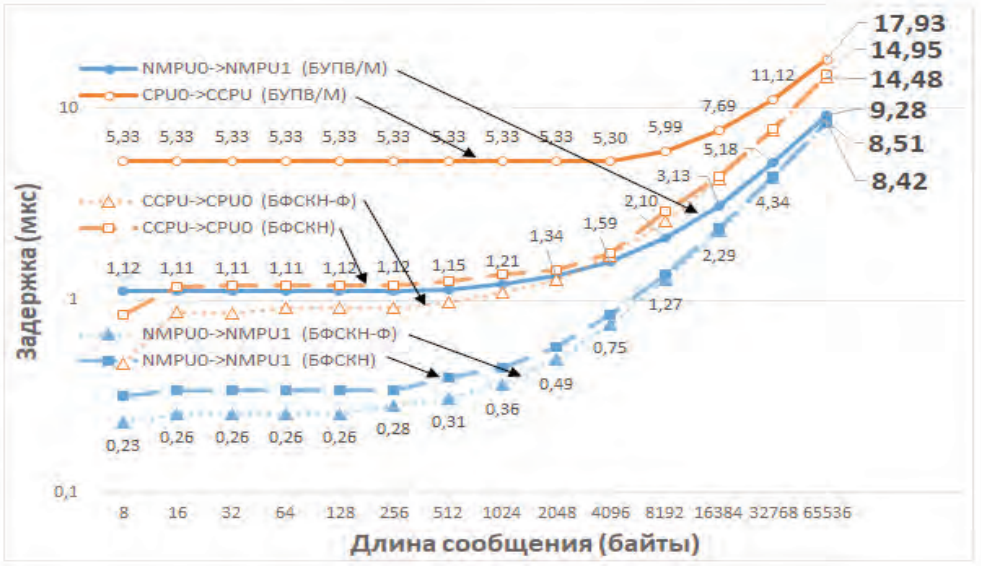

Рис.6. Задержки передачи сообщений между узлам через компорты с использованием разных библиотек

1. Задержки передачи сообщений для библиотек БФСКН-Ф и БФСКН близки соответственно для пересылок между NMPU-узлами и между центральным узлом и кластерным ARM-узлом.
2. Динамика роста задержек при увеличении длины сообщения для всех библиотек и всех пар узлов слабая (это очень хорошо!), это имеет место до сообщения в 1024 байта.

При пересылке из DDR-памяти кластера в DDR-память центрального узла через шину AXI в 640 MHz развивается пропускная способность в 78.8% от пиковой.

Данные рис.8 свидетельствуют о высокой реактивности запуска блока MDMAC – задержки передачи коротких сообщений составляют от 0.3 мксек до 0.6 мксек, причем до 1024 байт эта характеристика не

выходит за эти пределы, это очень сильный результат, поскольку неэффективная передача именно коротких сообщений – известная проблема параллельного программирования, она ограничивает степень распараллеливания программ.

Полученные высокие характеристики для библиотек параллельного программирования NM6408MP объясняются возможностями оборудования этого процессора, что позволило не использовать реализацию MPI с буферизацией сообщений и лишними их копированиями в памяти.

кластера – 254 нсек; глобальный барьер 21 ядра процессора – 580 нсек. Если сравнивать с многосокетными узлами [10, 11], то барьерных синхронизаций быстрее 1 мксек не встречается. Если же, к примеру, взять 2 таких узла (по 48 ядер на каждом), то при загрузке на каждом из этих узлов 16 ядер барьерная синхронизация выполняется для двух узлов за 12 мксек, а при загрузке 32 ядер на каждом узле – за 15 мксек.

Таким образом, преимущества по быстродействию и на барьерных синхронизациях библиотек NM6408MP очевидны.

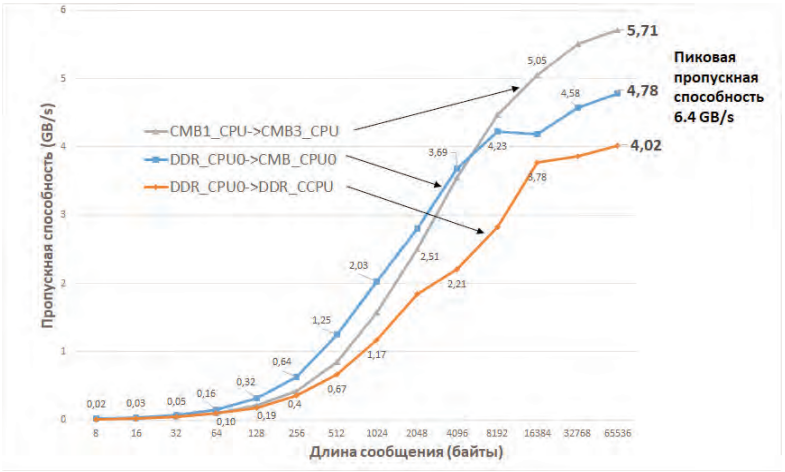

Рис.7. Пропускные способности передачи сообщений между узлам через MDMAC для БФСКН-Ф и БФСКН

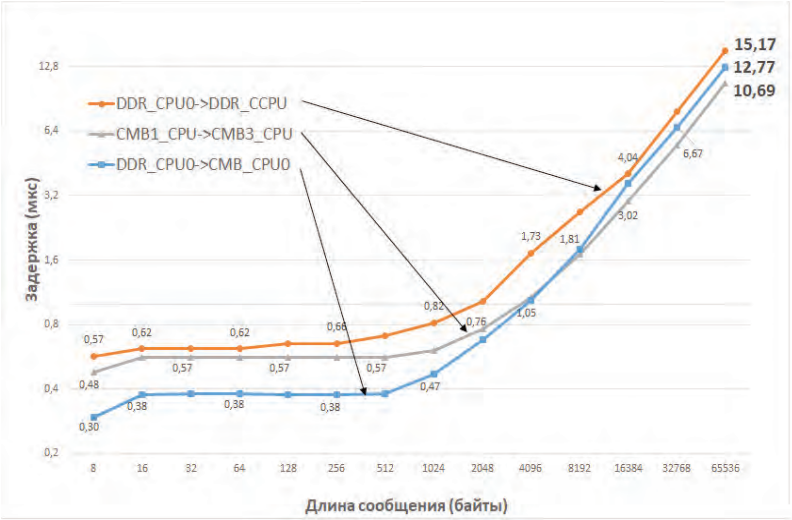

Рис.8. Задержки передачи сообщений между узлам через MDMAC для БФСКН-Ф и БФСКН

Барьерные синхронизации реализуются с использованием специальных регистров фиксации событий. Благодаря этому получаются малые времена барьерных синхронизаций: локальный барьер для пяти процессов

5. Заключение

В статье рассмотрены три библиотеки параллельного программирования для микропроцессора NM6408MP. На самом деле, этих библиотек было четыре. Разра-

ботке рассмотренных трех библиотек предшествовало создание опытным инженером, хорошо владеющим средствами программирования нижнего уровня для NM6408MP, экспериментальной библиотеки БУПВ с некоторыми принципиальными упрощениями реализации, но охватывающей все определенные для БУПВ функции. Допускалась некоторая некорректность реализации и потери по эффективности ради простоты реализации. Например, допускалось, что все синхропеременные для эксклюзивного доступа были расположены на уровне центрального узла.

Эта экспериментальная реализация впоследствии была тщательно протестирована на наборах функциональных и оценочных тестов, изучена коллективом изначально менее опытных специалистов, которые прошли таким образом обучение и реализовали потом рассмотренные в статье библиотеки, но по новым схемам и алгоритмам, которые сняли упрощения и повысили быстродействие. Это позволило поднять пропускную способность при передаче по сети с коммутацией каналов в 5-10 раз, в зависимости от длины сообщения, также резко снизить выполнение локальных и глобальных синхронизаций.

В настоящее время библиотеки БФСКН-Ф и БФСКН готовы к эксплуатации. Не исключено, что некоторые функции в них еще будут добавлены по желанию пользователей.

В ближайшее время основная работа предполагается по библиотеке БУПВ/М, как по ее комплексному тестированию, так и по развитию. Развитие БУПВ/М и введение ее в эксплуатацию предполагает следующие этапы: БУПВ/М1 – все возможности кроме X-маршрутов (июль 2019); БУПВ/М2 – добавление X-маршрутов и функций работы с host-процессором (август 2019); БУПВ/М3 – добавление EL-маршрутов, выход на работу с сетью NM6408MP (сентябрь-октябрь 2019); БУПВ/М4 – введение средств трассировки процессов, сервисных функций (ноябрь-декабрь 2019).

Наиболее важным в этом процессе видится выход на работу с сетью NM6408MP, сначала из 4-х процессоров (пиковая производительность 2 TFlops на FP32, 0.5 TFlops на FP64). Далее такими четверками эта сеть может произвольно наращиваться. Библиотека БУПВ/М обеспечит надежное выполнение параллельных программ на такой сети и удобную их разработку.

6. Выводы

Разработка библиотек показала, что для программистов – прикладников необходимы средства с разным балансом предоставляемых возможностей, удобства программирования и надежности. Тем не менее, представляется неразумным работать с процессами в многоядерном процессоре по физическим адресам ядер, на которых эти процессы находятся. Уровень библиотеки БФСКН видится тем, ниже которого опускаться не стоит.

Полученные высокие характеристики на освоенной в библиотеках внутренней высокоскоростной сети NM6408MP с коммутацией каналов опровергают сложившиеся с начала 2000-х годов представления, что внутренняя сеть процессора должна быть только сетью с коммутацией пакетов. Однако следует высказать пожелание, чтобы в этой сети все узлы были доступны, т.е. больше не возникала «проблема X-маршрутов».

Для удобства реализации синхронизации параллельных процессов желательно введение в будущих процессорах общепринятых атомарных операций с памятью, а не эксклюзивных операций чтения/записи, да еще с ограниченным количеством реализующих их мониторов, т.е. чтобы больше не было »проблемы мониторов».

Разработка отечественной реализации библиотеки типа MPI, хоть и сильно упрощенной, представляется событием в отрасли, поскольку обычно применяются американские версии реализации MPI, адаптированные посредством замены выделенного в них интерфейса на целевое оборудование.

Литература

1. Эйсымонт А.Л., Черников В.М., Черников Ан.В., Черников Ал.В., Косоруков Д.Е., Насонов И.И., Комлев А.А. Гетерогенная многопроцессорная система на кристалле с производительностью 512 Gflops // Системы высокой доступности, 2018, т.14, №3, с.49-56.
2. Адамов А.А., Павлухин П.В., Биконов Д.В., Эйсымонт А.Л., Эйсымонт Л.К. Альтернативные современным GPGPU перспективные универсальные и специализированные процессоры-ускорители // Вопросы кибербезопасности, номер 4, 2019, стр. 13-21.
3. Message Passing Interface, https://ru.wikipedia.org/wiki/Message_Passing_Interface
4. SHMEM, https://en.wikipedia.org/wiki/SHMEM
5. Chao Y. [et al.] A Novel DSP Architecture for Scientific Computing and Deep Learning // IEEE Access, Vol 7, April 2, 2019, pp 36413-36425.
6. William J. Dally W.J., Towles B. Route packets, not wires: on-chip interconnection networks // Proc. of the 38th conference on Design Automation (DAC), pages 684–689, 2001.
7. Dai G. [et al.] GraphH: A Processing-in-Memory Architecture for Large-scale Graph Processing // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems , Vol 38 , №4, April 2019, p.640-653.
8. Fan D.-R. [et al.] Godson-T: an efficient many-core architecture for parallel program executions // JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY, 24(6): 1061-1073, Nov.2009
9. Dally W., Balford J. [et al.] An Energy-Efficient Prpcessor Architecture // IEEE Computer Architecture Letters, 2008,v.7, №1, 4 pp.
10. Кудрявцев М., Эйсымонт Л., Мошкин Д., Полунин М. Суперкластеры — между прошлым и будущим // Открытые системы, №8, 2008.
11. Речинский А., Горбунов В., Эйсымонт Л. Суперкластер с глобально адресуемой памятью // Открытые системы, №7, 2011, с.21-25.
12. ARM Synchronization Primitives. Development Article. ARM, 2009, 18 pp.

THREE-LEVEL PARALLEL PROGRAMMING SYSTEM FOR THE HYBRID 21-CORE SCALAR-VECTOR MICROPROCESSOR NM6408MP

Bikonov D.V.[5], Puzikov A.D.[6], Sivtsov A.C.[7], Eisymont L.K.[8]

Abstract: The hybrid microprocessor NM6408MP (СБИС 1879ВМ8Я) with a peak performance of 0.5 TFlops for 32-bit floating-point numbers before the appearance of the Elbrus 8C was two years the most high-performance in the country. The processor contains 5 CortexA5 cores and 16 nmc4 scalar-vector cores with the NeuroMatrix architecture. The main high-level programming languages for the processor are C and C ++. The task was to develop programming tools for describing the communication of parallel processes and their synchronization. The article describes three libraries of different levels that solve this problem, taking into account the original features of the processor. The user interface of the functions of these libraries based on the style of the MPI and SHMEM libraries.

Keywords: MPI, SHMEM, channel-switched network, hierarchical on-chip memory, Send / Receive protocol, barrier synchronization, control of communication resources of the processor, exclusive memory access.

References

1. Eysymont A.L.. Chernikov V.M.. Chernikov An.V.. Chernikov Al.V.. Kosorukov D.E.. Nasonov I.I.. Komlev A.A. Geterogennaya mnogoprotsessornaya sistema na kristalle s proizvoditelnostyu 512 Gflops // Sistemy vysokoy dostupnosti. 2018. t.14. №3. s.49-56.
2. Adamov A.A.. Pavlukhin P.V.. Bikonov D.V.. Eysymont A.L.. Eysymont L.K. Alternativnyye sovremennym GPGPU perspektivnyye universalnyye i spetsializirovannyye protsessory-uskoriteli // Voprosy kiberbezopasnosti. nomer 4, 2019, str. 13-21.
3. Message Passing Interface, https://ru.wikipedia.org/wiki/Message_Passing_Interface
4. SHMEM, https://en.wikipedia.org/wiki/SHMEM
5. Chao Y. [et al.] A Novel DSP Architecture for Scientific Computing and Deep Learning // IEEE Access, Vol 7, April 2, 2019, pp 36413-36425.
6. William J. Dally W.J., Towles B. Route packets, not wires: on-chip interconnection networks // Proc. of the 38th conference on Design Automation (DAC), pages 684–689, 2001.
7. Dai G. [et al.] GraphH: A Processing-in-Memory Architecture for Large-scale Graph Processing // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems , Vol 38 , №4, April 2019, p.640-653.
8. Fan D.-R. [et al.] Godson-T: an efficient many-core architecture for parallel program executions // JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY, 24(6): 1061-1073, Nov.2009
9. DallyW., BalfordJ. [et al.] An Energy-Efficient Processor Architecture // IEEE Computer Architecture Letters, 2008,v.7, №1, 4 pp.
10. Kudryavtsev M.. Eysymont L.. Moshkin D..Polunin M. Superklastery — mezhdu proshlym i budushchim //Otkrytyye sistemy. №8
11. Rechinskiy A.. Gorbunov V.. Eysymont L. Superklaster s globalno adresuyemoy pamyatyu // Otkrytyye sistemy. №7. 2011. s.21-25.
12. ARM Synchronization Primitives. Development Article. ARM, 2009, 18 pp.

5 Dmitry Bikonov, Chief Specialist, Research Center « Module », Moscow, Russia. E-mail: d.bikonov@module.ru
6 Artem Puzikov, Software Engineer, Research Center « Module », Moscow, Russia. E-mail: a.puzikov@module.ru
7 Alexey Sivtsov, Senior Software Engineer, Research Center « Module », Moscow, Russia. E-mail: a.sivtsov@module.ru
8 Leonid Eisymont, Scientific – PhD, Consultant, Research Center «Module», Moscow, Russia. E-mail: verger-lk@yandex.ru